Learning the Popularity of Items for Mobile Tourist Guides

Patrick Hiesel
Technical University of Munich
Boltzmannstraße 3
85748 Garching, Germany
hiesel@in.tum.de

Matthias Braunhofer
Free University of
Bozen-Bolzano
Piazza Domenicani 3
39100 Bolzano, Italy
mbraunhofer@unibz.it

Wolfgang Wörndl
Technical University of Munich
Boltzmannstraße 3
85748 Garching, Germany
woerndl@in.tum.de

ABSTRACT

A context-aware recommender system incorporates the knowledge of different contextual factors such as time or weather information to improve item suggestions made to a user. This requires the system to have a large knowledge base for inferring contextual information and enabling accurate and timely recommendations. We present a versatile approach for a context-aware recommender system in the tourism domain by crawling publicly available information from a variety of sources and learning the contextual popularity of points of interest based on a generalized check-in model. We have deployed a test instance of our system for the greater area of Munich and the German state of Bavaria. Analyzing the results from the offline learning has led to interesting insights including when and in which weather conditions certain items are popular.

CCS Concepts

•**Information systems** → **Recommender systems;**

Keywords

Recommender systems; context; data analytics; mobile guides

1. INTRODUCTION

Recommender systems are a composition of software tools and techniques that suggest items to users that are likely to be interesting to them and relevant to their needs [10]. Traditional recommender systems consider items liked/rated by users in the past and possibly some additional information such as item characteristics to estimate the ratings for items that the users have not yet consumed [1]. Applications range from suggesting products that have been bought together by other users in the past over suggesting people a user might know based on their existing list of friends to suggesting music based on genres listened to.

Context-aware recommenders enhance traditional recommender systems by incorporating the knowledge of different contextual factors - such as time or weather information - to further improve the item suggestions made to a user [1]. These systems seek to better match users and their current context with items that are popular in the same or similar contexts.

One major application area of context-aware recommender systems is travel and tourism, where scenarios are significantly more complicated than traditional user-product matchings [3]. In addition to the general preferences of a user, the utility and relevance that a point of interest (POI) has to a user heavily depends on the user's current context. A beer garden, for instance, would provide a higher value to the user on sunny summer days rather than on rainy winter days and a car that knows a driver's route, fuel level and gas prices can make better suggestions for gas stations to refuel. This is especially important in the scenario of a proactive recommender system [14], i.e., a recommender that pushes item suggestions to the user based on the current situation (e.g. location, time of day or weather) without explicit user request.

In our research, we propose, implement and evaluate a novel approach for a context-aware recommender system in the tourism domain by aggregating publicly available information from a variety of sources and learning the contextual popularity of POIs based on a generalized check-in model. We aggregate different types of data, including POIs, check-ins and contextual information to build a knowledge base and infer knowledge about the contextual popularity of items focussing on aspects of temporal and geographic context. The gained knowledge can also be utilized to mitigate cold-start problems when no or little information about the user is available.

In the following, we first explore related work (Section 2), describe the context model, data sources and system design of our implementation (Section 3) and then present the process and results of analyzing the data (Section 4). Finally, Section 5 draws conclusions and discusses open future work directions.

2. RELATED WORK

Context-aware recommender systems have been a topic of growing research interest in the recent years and aim at generating more relevant recommendations by adapting to the specific contextual situations of the user and the recommended items (e.g., weather, temperature, season and mood) [1]. There exist numerous commercial and research systems, such as Foursquare, Yelp, South Tyrol Suggests (STS) [4] and ReRex [2], that have already been successfully implemented and that exploit the current user's and item's context when recommending items. These systems

RecTour 2016 - Workshop on Recommenders in Tourism held in conjunction with the 10th ACM Conference on Recommender Systems (RecSys), September 15, 2016, Boston, MA, USA.

use different approaches to incorporate context into the recommendation process. Roughly, these approaches can be divided into three categories [1]: (i) contextual pre-filtering, where context is used for selecting the relevant set of ratings before computing predictions with a traditional, two-dimensional prediction model; (ii) contextual post-filtering, where context is used to adjust the recommendation list resulting from a two-dimensional rating prediction model; and (iii) contextual modeling, where context is directly incorporated into the prediction model.

Most current context-aware recommender systems work in pull mode, i.e., the user has to explicitly make a request (pull) for recommendations, possibly by entering information about her preferences, needs and constraints. A new generation of context-aware recommender systems, called proactive recommender systems, are instead pushing recommendations to users without their specific request, when they are in a contextual situation that the system considers as suitable for the recommendations [14]. Despite the advantages of proactive recommender systems - especially in mobile usage scenarios - relatively little research has been conducted specifically on this topic. One example is [14], where the authors proposed a proactive recommender system model consisting of two phases: (i) the situation assessment phase, which evaluates whether or not the current contextual situation calls for a recommendation; and (ii) item assessment phase, which is only executed when the first phase indicates a promising situation and assesses the candidate items to finally decide which items should be pushed to the user as recommendations. Subsequent work in [13] has evaluated the effectiveness of the proposed model by applying it to a restaurant recommender system, and found that users highly appreciate proactive recommendations if they are relevant and properly timed.

In another work, Dali et al. [5] presented different timing models based on random forest to classify user contexts that are suitable for recommendations and user contexts in which users are highly likely to refuse any recommendations. Results from a user study revealed that a hybrid model that first decides whether it should use a personal or non-personal timing model, and then classifies whether the context is suitable for recommendations is superior to both the personal or non-personal timing models.

In another study, Pielot et al. [9] showed that boredom can be inferred from patterns of mobile phone usage and that users are more likely to appreciate proactive recommendations during inferred phases of boredom. Hence, they concluded that using boredom as trigger independent from content might help to make proactive recommendations a more pleasant experience for users.

Finally, Borrís et al. [3] survey intelligent recommender systems in travel and tourism and also mention context and proactivity as important factors.

3. CONTEXT, DATA SOURCES AND SYSTEM DESIGN

In order to create a versatile system that can gather relevant data for any geographic area and infer the contextual popularities of POIs, we first discuss the context model that our work is based on. This leads to the features we need to extract from the data we crawl. We then discuss different data sources for items, check-ins and context and present a brief system design of both the backend implementation and a corresponding mobile client.

3.1 Context Model

Our research is based on a context model for proactivity in mobile recommender systems defined by Wörndl et.al. [14]. The model relies on domain-dependent context modeling in four distinct categories: *user context, temporal context, geographic context* and *social context*. The recommendation process itself is divided into two phases to first analyze the current situation and then examine the suitability of particular items. This allows for both a proactive and passive recommender system.

The context model defines the data we need to aggregate and the features that should be extracted. In particular, we want to infer the temporal and geographic context of items. The user context is extracted directly in our mobile app prototype in a later stage. The social context is neglected in our prototype but could be integrated in our approach. Based on this context model, we focus on the following contextual features: *weather condition, temperature, season, day of week, time of day* and *time of year*.

3.2 Data Sources

The overall goal of this work to recommend POIs in a mobile tourist guide based on publicly available data sources. We therefore need to acquire and aggregate data in three categories: items (the POIs), check-ins (to determine the popularity of items) and context.

3.2.1 Items

The POIs form the foundation of our system as the user's overall perception of the system first and foremost depends on the quality and suitability of the recommended items. We therefore designed a general model for an item in our domain that can then be populated with data. Core data fields include: name, description, location, images, phone, rating, street, city, country. While the core fields should always be populated, each data source's crawler can define additional fields it wants to persists. Based on this model, we examined different, publicly accessible data sources: Foursquare, Yelp, Quermania, Facebook, Wikipedia, Open Street Maps. We designed and implemented crawlers for each of those sources and aggregated over 175,000 items for the German State of Bavaria.

3.2.2 Check-ins

Based on this knowledge base of 175,000 items, we want to infer the popularity in accordance to the context model. Our premise is that a place is popular if many people visit it. It follows that a place is popular in a certain context (e.g. on a sunny Saturday evening in summer) if many people visit it when this context condition applies.

To infer this knowledge, we must first define a model for determining how popular a place is at a given time or any other context. We base this model on a generalized check-in. In our model, a check-in can be any evidence that a user visited a place at a certain time. This includes an explicit check-in on Foursquare or Facebook, as well as implicit check-ins by taking a picture or generating a GPS trace on a smartphone. A check-in marks a singular point in time. To make an educated judgement on how many people are present at any point in time, we must further make an assumption about the average duration of a visit. There has been research on the activity duration of different activities [8] that we use to infer the number of present users from check-ins at any point in time based on the POI type.

We looked at different publicly available data source to crawl check-ins from including Flickr, Twitter and Foursquare. Flickr is a photo sharing platform with a rich set of geo-tagged photos. A geo-tagged photo contains the latitude and longitude of the place it was taken as well as a level of accuracy of this information. In addition, a textual description is provided for most of the images. We used Flickr as the primary source of check-ins for rural areas, establishing a place-to-check-in mapping in a sparsely populated area which is more straightforward given the geo coordinates of both places. In our research, we have sampled about 249,000 images to be used as check-ins.
Twitter offers a large number of tweets that can be associated with a place and hence counted as a check-in. There are two main features, that can be used for association: geo-tagged tweets and Twitter Places. Geo-tagged tweets carry a latitude and longitude and can be associated to places using geo-fencing. While this is a valid approach in rural areas, it is insufficient for cities due to the number of places being close to each other. Twitter places specify a specific geographic place, such as a city or a restaurant and can be mapped to a POI directly. In our research, we have sampled 29m tweets in about eight weeks using Twitter's Streaming API.
Foursquare has an explicit check-in feature - similar to Facebook - that would let users check-in at a place using a button in their app. In addition, Foursquare samples the user's location on the smartphone to proactively recommend POIs. While this data seems promising, none of it is publicly accessible and was therefore not used for our prototype. On the contrast, some users link their Twitter account with their Foursquare account resulting in each explicit check-in triggering a public tweet. Following the approach proposed by Melia and Segui [8], we extracted this information from the tweets we collected and linked them to Foursquare venues in our database. Based on our Twitter dataset of 29m tweets, we successfully linked 2.9m tweets to a Foursquare Swarm POI. The association rate for the German state of Bavaria was 8 check-ins (tweets) per 1,000 POIs and enriched our dataset for cities.

3.2.3 *Context*

Based on the aggregated knowledge base of items and check-ins we now want to add a third data source to enrich our data with context. We hereby focus primarily on dimensions of the geographic and temporal context. Given the timestamp of a single check-in we can infer all attributes of the temporal context using a static calendar library. We therefore use java.util.Calendar to infer *season, day of week, time of day* and *time of year*. To infer the weather condition and temperature of check-ins we added Wunderground's Weather History API that can be used to provide the weather for a place at any given time in the past. Using these two sources, we were able to infer the temporal and geographic context for all check-ins.

3.3 System Design and Server Implementation

We implemented our prototype in Scala using the Play! framework. Scala is a language that is executed on top of the Java Virtual Machine and is fully interoperable with Java. The language combines object-oriented programming with functional programming which makes operations on arrays and lists easy to implement. On top, all Java libraries are usable in Scala which is a major benefit. The Play! framework offers a modern web MVC architecture with a simple control flow and the advantage of using templates in the views. We have combined this with Slick - a functional relational mapper - to have a persistent database layer based on a performance-tuned MySQL instance. Slick offers a clean way of accessing and filtering persistent data sets using Scala's functional API.

Figure 1: Screenshot of the Mobile App with the Popularity Graph for Time of Day

Data aggregation is either done through a REST API (if available) or through one of our crawlers. In the course of this research, we have created multiple crawlers and spiders on top of JSoup to extract knowledge from publicly available sources. We used Amazon AWS resources to carry out parts of these tasks with on-demand resources, while maintaining only one dedicated server at all times. We used SQS, a distributed pipe service, to communicate between the different servers.

3.4 Mobile Application Design

We have also designed, developed and tested a user interface concept to investigate how to communicate contextual information about recommended items to the user in a mobile tourist guide [7]. Thereby, the user can retrieve information about interesting POIs and review various graphs about the item popularity in an item detail screen. We show a textual summary of the popularity peak for both weather and day/time. We also render line and bar charts to show the popularity across different context dimensions, such as day of the week, time of day (see Figure 1 for an example) and season. The mobile application was implemented using the cross-platform framework Ionic and is fully functional.

To receive early feedback for our concept, we evaluated our system in a user study with 14 subjects. The participants were asked to test the application for two weeks and then complete a survey about their experience with the user interface elements. The study results indicated that the pop-

ularity inference and graphs provide benefits for users. Users stated that popularity graphs assisted them while deciding which place to visit. More detailed results of this preliminary study can be found in [7].

The focus in this paper is not on the user interface but on how to analyze the collected data and infer insights that can then be integrated in a mobile tourist guide. We present our course of action and the gained results of the offline learning in the next section.

4. OFFLINE ANALYSIS AND PROBABILITY LEARNING

To infer basic popularities for different conditions we follow the data analytics process proposed by Runkler [11]. The process decomposes the data analytics pipeline into four steps: preparation, preprocessing, analysis and post-processing. Following this process, we have identified the following tasks for each step:

1. Preparation: identifying goals and research question; data collection.
2. Pre-processing: merging POIs from different sources; associating POIs with check-ins filtering, sampling and discretization, normalization.
3. Analysis: visualization, popularity inference and prediction.
4. Post-processing: evaluation.

To follow this process, we define an overall research question and subquestions that we seek to answer.

- Overall research question: how can we learn (infer) the popularity of POIs for a context-aware recommender system?
- Subquestion 1: how can POIs from different sources be matched and duplicates be eliminated?
- Subquestion 2: how can POIs be associated with check-ins?
- Subquestion 3: given a set of check-ins for a POI, how can we infer the popularity under different temporal conditions?
- Subquestion 4: given a set of check-ins for a POI, how can we infer the popularity under different geographic conditions?
- Subquestion 5: given a set of check-ins for a POI and the base popularities, which algorithms are suitable for making a compound decision/recommendation?

In the following, we present our approaches and results on these tasks according to the subquestions we are trying to answer.

4.1 Merging POIs from Different Sources

We have added different POI sources to our system, including Foursquare, Yelp and Quermania. Especially Foursquare and Yelp provide a lot of overlapping data, since both have restaurants and bars in their database. This leaves us with the problem of identifying and merging co-referent POIs, as we do not want to show or recommend the same POI multiple times.

Merging co-referent POIs has been extensively studied and there are multiple approaches to the problem, e.g., using a fuzzy set and probability theory [12] or a DBSCAN, a common clustering algorithm [6]. The fuzzy set approach assumes, that the majority of POIs are user-generated and therefore has large differences in between two versions of a POI's name. Our data differs from the assumptions of these papers in a way, that it is already pre-filtered by Yelp and Foursquare. We therefore take a simple approach matching POIs from different sources: we compare the Levenshtein distance of the names of the two POIs in question and investigate their geographic distance.

4.2 Associating POIs with Check-Ins

Associating POIs with check-ins is a non-trivial task, given that the data comes from entirely different sources. Depending on the source of the POI and the check-in, we have identified several ways of creating an association:

4.2.1 POIs with Flickr/Twitter Check-ins

Flickr and Twitter both provide data records that have coordinates, a timestamp and some textual data like the tweet's content or the Flickr image's caption. Associating these records with a POI presents a hard problem that can be tackled in many different ways:

Simple Geofencing: The computationally easiest option is to use a rectangular geofence around a POI and associate each check-in record within this area with the POI. This works well for exposed POIs (i.e., satellite POIs that have no other POIs around them), but is inapplicable to the majority of POIs in cities including restaurants and bars.

Advanced Geofencing: To improve both the FPR (false positive rate) and FNR (false negative rate) of the simple geofencing approach, we propose a more complex way of setting up the geofence. Depending on the POI's type, a more complex geofence structure can range from a smaller circle (suitable, e.g., for bars or restaurants in cities) to a polygon following the shape of ski slopes or trails.

Clustering: can be done in a supervised, unsupervised or semi-supervised fashion. A supervised clustering approach would assume that we have a couple of check-ins per POI where we have obtained evidence that they belong to this POI. Other nearby check-ins could be associated using algorithms like KNN (see Evidence-based Clustering below). An unsupervised approach could use algorithms like DBSCAN to discover clusters in unclassified data. Once the algorithm has discovered clusters, we could use different techniques to associate POIs with clusters.

Evidence-based Clustering: As mentioned with simple clustering, one approach could be to use supervised or semi-supervised clustering for association. To start this algorithm we would need a base dataset of check-ins that are associated with POIs. One approach to generate such a dataset would be to use evidence from the metadata. Most tweets/Flickr descriptions contain the topic of the tweet or image such as "Neuschwanstein". Using simple word-matching or more complex NLP techniques we would obtain a base set of check-ins to use for clustering and association.

We use Flickr and Twitter check-ins for exposed sights to deliver a proof of concept for our pipeline and model and could therefore use simple geofencing to associate check-ins with POIs (by a square whose size depends on the POI type). Our approach has linked 280,983 check-ins with 177 sights. While this approach was sufficient for our use case, future

work could lie in exploring other association techniques to make using these check-in sources viable for more densely populated areas.

4.2.2 *Foursquare POIs to Check-ins using Twitter*

As briefly outlined before, Melia and Segui [8] proposed an approach to aggregate public Foursquare (Swarm) check-ins using Twitter. Foursquare users that have linked their account with Twitter will automatically publish a tweet if they check-in publicly. By analyzing the corresponding tweets using our data processing pipeline, we obtain the unique Foursquare ID of the POI where the user checked-in. In addition, we obtain the timestamp of the tweet and hence the check-in.

This approach is promising, as it established a definitive relation between a check-in and a POI (i.e., there are no false-positives). We have aggregated a mixed data set of tweets containing both geo-located tweets and tweets linked to public Foursquare and Swarm check-ins using the streaming API over eight weeks. The dataset has 29m tweets in total, 2,9m of which we have successfully linked to a Foursquare or Swarm POI.

Our prototype only incorporates POIs in the German State of Bavaria. We could therefore only use a small subset of the 2.9m tweets that link to a POI in our region. We could match tweets and POIs at a rate of 8 check-ins (tweets) per 1,000 POIs. This number is relatively low compared to the effort it took to obtain the data using the pipeline we outlined earlier. While this was not an ideal fit four our use case, the approach can be a better choice in regions with higher Foursquare adoption (such as New York or San Francisco).

4.3 Filtering, Sampling and Normalization

Until this point, we have crawled POIs, check-ins and context and associated POIs with check-ins. The next step is to filter the dataset to increase its quality, decide on sampling for context parameters and perform normalization if needed.

Filtering: For the qualitative evaluation and our mobile app prototype we focus on exposed sights out of our large POI dataset and have identified 176 sights with a high POI to check-in matching having both a low FNR and FPR. Furthermore, we filter out POIs that have less then 400 check-ins (on average each POI has 1570 check-ins) to retain a good accuracy for all contextual popularities. All contextual popularity groups partition the remaining check-ins into a maximum of 7 groups yielding 57 check-ins on average per group. This filtering gives us a set of 114 sights in the greater area of Munich and the German State of Bavaria that we use for further processing.

Sampling: The next step to aggregate the popularity is to define what dimensions should be explored and if a dimension is continuous or discrete. We hereby explore temporal and geographic dimensions separately. Discrete variables are easy to handle when it comes to inferring the popularity, as they can be represented by a fixed number of buckets (e.g., seven for *Day of Week*) and assigning check-ins to buckets is trivial. Continuous variables however, are more difficult as questions like "How popular is this POI at 10am?" can not be holistically answered using sampling and buckets, since when settling on a fixed bucket size (e.g., half an hour) the question "How popular is this POI at 10:11am?" can not be answered accurately. When decreasing the bucket-size, buckets will only have a few check-ins as a check-in only marks a specific timestamp making the popularity impossible to infer.

We therefore explored related work on activity duration [8] to assign a presence window to each check-in accounting for the time a user was present at the POI. The activity duration depends on the POIs category and ranges from 7:43 min for breakfast to 19:01 min for dinner. Parks and outdoor have a mean activity duration of 11:21 min. Given this knowledge, we set a bucket-size of 5 min and count the check-in towards three buckets for sights. In this approach we model a user's presence at a given time by adding 1 to three buckets. With this information, we can accurately answer questions in the form of "How popular is this POI at 10:11am?", as we have a bucket ranging from 10:10:00am to 10:14:59am. However, with this model we neglect the uncertainty at the beginning and end of the activity interval. The source of this uncertainty is the fact, that we only know one timestamp and infer the user presence window through (assumed) activity durations yielding a high uncertainty at the beginning and end of this interval. An alternative modeling approach could be to use Gaussians to augment presence intervals. This way, we can model the uncertainty in a statistically correct way.

We regard the Gaussian modeling as future work and base our presence interval on activity durations without augmenting Gaussians. We introduce *feltTemperature* as a discretization of the continuous temperature as this simplifies inference and makes our predictions easier to understand for users. Table 1 shows our set of inferred contextual variables for both the geographic and temporal context.

Variable	Type	Domain
dayOfWeek	D	{Mon,...,Sun}
timeOfDay	C	[0,23]
season	D	{Spring,Summer,Fall,Winter}
feltTemperature	D	{Hot,Warm,Pleasant, Mild,Cold,Freezing}
weatherCondition	D	{Sunny,Cloud,Rainy,Snowy}

Table 1: Temporal/Geographic Dimensions for Sampling. D=Discrete, C=Continuous

Normalization: Figure 2 shows an aggregation of all of our check-ins on a weekly basis using the presence window approach based on a presence window of 1 hour. One can clearly see from the graph, that about twice as much check-ins are made during weekends, than there are on a weekday. We have therefore thought about normalizing the data as a whole to have the same relative amount of check-ins per discrete bucket (e.g. Mon-Fri).

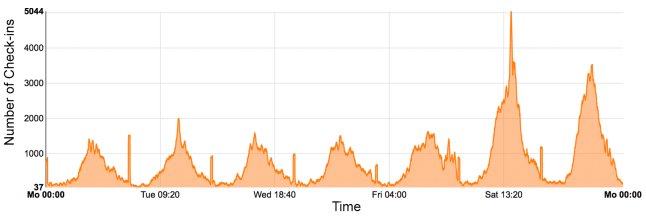

Figure 2: All Check-ins Aggregated on a Weekly View

After investigating our data in close detail, we decided against normalization as users tend to visit more POIs on

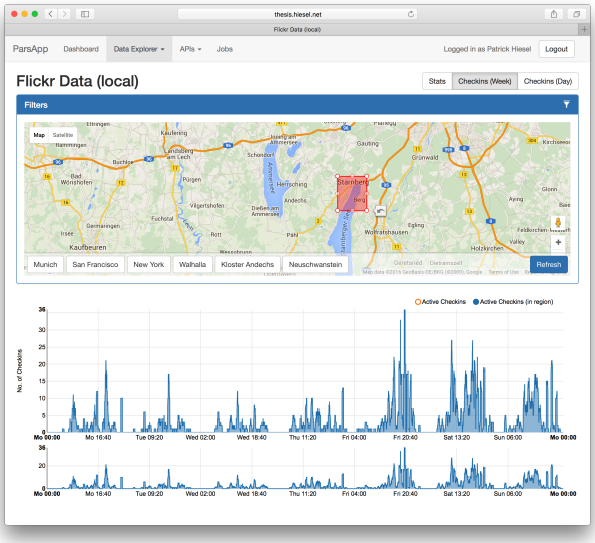

Figure 3: Geofilter Analysis

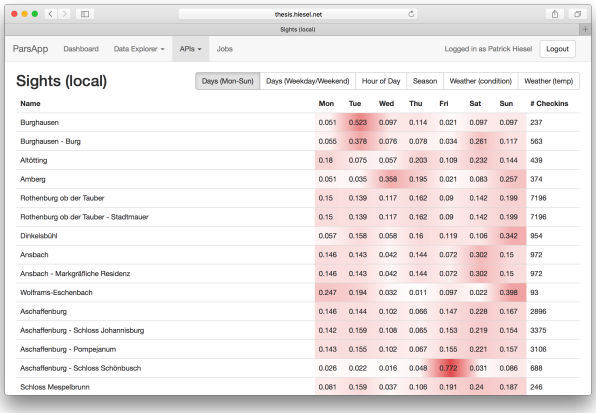

Figure 4: Popularity

the weekends/evenings when they spend leisure time. We therefore anticipate, that the weekend cliffs (as well as some other data patterns) are correct and beneficial for the sake of popularity inference.

4.4 Visualization, Popularity Inference and Prediction

4.4.1 *Visualization*

To effectively visualize our data, we created multiple analysis tools based on our server-side software. For effective analysis of the check-in data we build a tool to visualize different features of geo-based check-ins without an association to any POI. Our tool, as depicted in Figure 3, is able to apply a geofence to filter check-ins and compare the selected area to a baseline of all known check-ins which makes it easy to spot interesting patterns. Figure 3 shows all check-ins on a weekly level for the northern end of Starnberger See. This is a good example for the effectiveness of our tool, showing that this part of the lake is popular on Friday afternoon as well as on the weekend.

After an initial analysis using these tools, we started to model the popularity inference. For this purpose, we created different views, such as the one depicted in Figure 4, to visualize the inference outcome. Our view shows the probabilities (popularities) under different temporal and geographic conditions highlighting them with a gradient in red color such that one can easily spot patterns in the data.

4.4.2 *Popularity Inference*

Until this point, our analysis and visualizations yielded promising patterns. We now take a probabilistic approach to infer popularities for different contextual situations. The outcome we seek is the popularity P of a POI given a context C:

$$p(P|C) \tag{1}$$

When thinking about context and different dimensions as described before it would be most beneficial to split up C into temporal and geographic dimensions and their respective parameters (temperature t, weather condition wc, season s, weekday wd, etc.). This yields probabilities of the form:

$$p(P|t),\ p(P|wc),\ p(P|s),\ p(P|wd) \tag{2}$$

Having these base popularities would allow for compound calculations yielding the probability for a visit V to a certain POI given a situation with fixed contextual parameters:

$$p(V|C,P) \tag{3}$$

The concept of a visit is that given that a user will certainly visit a POI, what is the probability that the visit will happen at the context C.
There are different approaches to model the statistical dependence of context factors like weather condition and temperature (e.g., it does not snow when having 20°C) including Bayesian networks. While a full Bayesian model can increase accuracy, it requires a high degree of domain knowledge about the data and all parameters that is hard to obtain. We therefore use summation to infer the plain popularities for each context parameter:

$$p(P|wd = Monday, V = Walhalla) \tag{4}$$
$$= p(wd = Monday|V = Walhalla) \tag{5}$$

$$p(wd = Monday|V = Walhalla) \tag{6}$$
$$= \frac{\sum_{CheckinsForPOIOnMonday}}{\sum_{CheckinsForPOI}} \tag{7}$$

This approach yields a probabilistic distribution for each POI and each context dimension as depicted in Figure 4 which lets one judge under which conditions a place is popular. We can use prediction and the Bayes' theorem to make compound recommendations.

4.4.3 *Prediction*

With prediction or recommendation, we seek to answer the question "Where should I go (given the current timestamp and weather)?". With respect to our model, this would mean: what is the probability (or score) that I should choose a POI for my visit V given the inferred popularities of all

POIs P and the current context C:

$$p(V|C,P) \tag{8}$$

We propose two different approaches to answer this question: weighted additive scoring and using the Bayes' theorem.

Weighted Additive Scoring.

The first approach to obtain a popularity score given a context situation C is to use weighted additive scoring.

$$S(C,V) = \sum_{i \in [wc,t,d,s,h]} \alpha_i \; p(i,V) \tag{9}$$

For each POI we add the popularity for a specific set of context dimension (weather condition wc, temperature t, day d, season s, hour h) that we want to predict a score for. Furthermore, we multiply each dimension by a weighting factor α to make the model flexible. While we used static weights, future work could include learning of α through online of offline learning techniques.

While this model works well, it does not respect the different number of check-ins between POIs. Thus a POI with only 400 check-ins might outperform a POI with 4,000 check-ins, as the popularity values are better, while - in absolute visitor numbers - the second POI outperforms the first. This issue is addressed by an approach using Bayes' theorem.

Bayes Theorem.

Bayes' theorem can be used to inverse a dependent probability. Using summation to obtain the base probabilities as outlined in the previous section yields probabilities in the form of:

$$p(wd = Monday|V) \tag{10}$$

In other words, this means: "given that I visit a POI, what is the probability I would visit it on Monday?" (or: "what is this POI's Monday popularity"). When recommending items, we would like to answer questions of "It is Monday, which POI should I visit?":

$$p(V|wd = Monday) \tag{11}$$

So for a concrete POI (Walhalla) we can use Bayes' theorem to get the probability for a visit given our limited set of POIs and the premise that the user will visit one of these:

$$p(V = Walhalla|wd = Monday)$$
$$= \frac{p(wd = Monday|V = Walhalla)p(V = Walhalla)}{p(wd = Monday)} \tag{12}$$

All of the parameters from this model can be retrieved from our dataset:

$$p(wd = Monday|V = Walhalla) \tag{13}$$

(as obtained in the previous section)

$$p(V = Walhalla) = \frac{\sum_{CheckInsAtWalhalla}}{\sum_{AllCheckIns}} \tag{14}$$

$$p(wd = Monday) = \frac{\sum_{AllCheckInsOnMonday}}{\sum_{AllCheckIns}} \tag{15}$$

We found, that using a Bayesian approach disproportionally favors POIs that have a high number of check-ins even when being relatively unpopular and therefore used weighted additive scoring.

Using either one of those strategies leaves us with a probability (popularity) for each POI in the database, which can be used for ranking in combination with a standard weighted additive scoring approach for recommending a POI based on the different context factors.

4.5 Evaluation of Post-Processing

We discussed different types of evaluations for this part of our research with peers from the field of machine learning. While machine learning and data analytics applications are usually evaluated using cross-validation or any other sort of quantitative offline evaluation, we did not see a fit for these types of evaluations given our data and inference schema. We therefore decided to first evaluate our research by qualitatively assessing inferred popularities for selected results in a case study and discussing interesting findings and patterns. The ultimate goal is to integrate the results about item popularities into the mobile application (see Subsection 3.4) and conduct a large-scale user study to get real feedback.

Inferred Seasonal Popularity.

The inferred seasonal popularity performed best among all our inferred parameters. An excerpt of the POI data with seasonal popularity is shown in Table 2. While cities like Bad Tölz or Nürnberg are equally popular throughout the year, other sights peak in certain seasons. Kehlsteinhaus for instance - a tea house built for the Third Reich government that has been turned into an exhibition - is closed during spring and winter (November - April), as the road can not be maintained as soon as there is snowfall. It can be easily seen that this shutdown persists in the data making the sight less popular during winter and spring. Other places for outdoor activities show a clear tendency towards the warmer periods of the year, including Walchensee (being popular during Summer and Fall), Starnberger See (Summer) and Königssee (Summer and Fall).

POI Name	Spring	Summer	Fall	Winter
Bad Tölz	0.215	0.202	0.215	0.368
Nürnberg	0.238	0.242	0.295	0.226
Walchensee	0.076	0.456	0.291	0.177
Kehlsteinhaus	0.044	0.572	0.35	0.034
Starnberger See	0.003	0.932	0.045	0.019
Kloster Andechs	0.314	0.253	0.365	0.067
Königssee	0.193	0.367	0.305	0.135

Table 2: Excerpt of Inferred Seasonal Popularity: $p(V|s,P)$. $\mu = 1,660$ Check-ins

Inferred Weather Condition Popularity.

The weather condition overall has a clear tendency towards sunny weather, with more then 90% of the dataset having a sunny popularity of 0.5 or higher. Nonetheless, there are still some patterns to get indications if the inference technique is correct. While cities like Munich are within our average (mostly sunny), places for outdoor activities like Starnberger See, Andechs or Almbachklamm have a notably higher sunny popularity. Table 3 shows an excerpt from inferred weather condition popularities.

POI Name	Sunny	Cloudy	Rainy	Snowy	Unkn.
Munich	0.532	0.138	0.324	0.005	0.001
Andechs	0.711	0.072	0.209	0.006	0.002
Roseninsel	0.796	0.127	0.072	0.005	0
Almbachkl.	0.907	0	0.093	0	0

Table 3: Excerpt of Inferred Weather Condition Popularity: $p(V|wc, P)$. $\mu = 1,750$ Check-ins

5. CONCLUSIONS AND FUTURE WORK

In this work, we have designed and implemented a data analytics process to infer the popularity of POIs for a context-aware recommender system using publicly accessible data from various data sources. We have elaborated different approaches on individual subtasks, e.g., applying probabilistic modeling to learn the popularity for POIs in different contextual situations such as *weather condition, day of week* and *time of day*. We qualitatively evaluated our approach in a case study and also implemented a corresponding mobile application with a brief preliminary user study [7].

In addition to the explained results, we discovered other useful purposes for the dataset and the inferred information. For example, to determine and visualize popular areas on a map. This could be useful for visitors to find spots for taking iconic pictures or maybe find a less-crowded spot and avoid the most popular areas. To do so, we filtered our data by first selecting the check-ins associated with a POI and then applying another filter to eliminate check-ins with an accuracy worse then 10 meters. Then this filtered dataset is fed into a heatmap algorithm for visualization using Google Maps' heatmaps extension. Figure 5 shows popular (photo) spots around Neuschwanstein Castle as an example.

Future work includes integrating the item popularities into the mobile tourist guide application and proactively recommending POIs based on the current user context. We then plan to conduct a larger user study to investigate whether this approach leads to useful and timely recommendations from a user's perspective in a real setting. Another area for improvement is not to solely rely on explicit user check-ins but also utilize tracking data from smartphones.

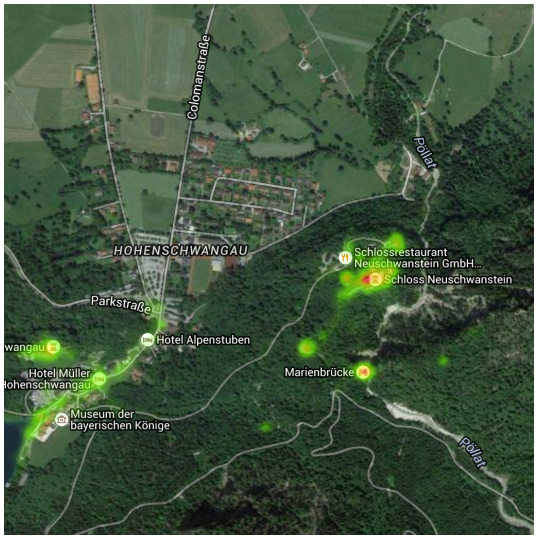

Figure 5: Heatmap for Neuschwanstein Castle Based on Check-in Data

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 191–226. Springer, 2015.

[2] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526, 2012.

[3] J. Borrís, A. Moreno, and A. Valls. Review: Intelligent tourism recommender systems: A survey. *Expert Syst. Appl.*, 41(16):7370–7389, Nov. 2014.

[4] M. Braunhofer, M. Elahi, and F. Ricci. Usability assessment of a context-aware and personality-based mobile recommender system. In *E-commerce and web technologies*, pages 77–88. Springer, 2014.

[5] N. Dali Betzalel, B. Shapira, and L. Rokach. Please, not now!: A model for timing recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 297–300. ACM, 2015.

[6] M. Daszykowski and B. Walczak. Density-Based Clustering Methods. *Comprehensive Chemometrics*, 2:635–654, 2010.

[7] P. Hiesel, W. Wörndl, M. Braunhofer, and D. Herzog. A User Interface Concept for Context-Aware Recommender Systems. In *Mensch und Computer 2016 Tagungsband*. De Gruyter, 2016.

[8] J. Melià-Seguí, R. Zhang, and E. Bart. Activity duration analysis for context-aware services using foursquare check-ins. In *Proceedings of the 2012 international workshop on Self-aware internet of things*, pages 13–18, 2012.

[9] M. Pielot, L. Baltrunas, and N. Oliver. Boredom-triggered proactive recommendations. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pages 1106–1110. ACM, 2015.

[10] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer Verlag, New York, NY, USA, 2010.

[11] T. A. Runkler. *Data Analytics: Models and Algorithms for Intelligent Data Analysis*. Springer, 2012.

[12] G. D. Tre and A. Bronselaer. Consistently handling geographical user data: Merging of coreferent POIs. In *Proceedings of the 2010 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*, 2010.

[13] D. G. Vico, W. Wörndl, and R. Bader. A study on proactive delivery of restaurant recommendations for android smartphones. In *ACM RecSys workshop on personalization in mobile applications, Chicago, USA*, 2011.

[14] W. Wörndl, J. Hübner, R. Bader, and D. Gallego-Vico. A model for proactivity in mobile, context-aware recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 273–276. ACM, 2011.